```
df=pd.read_csv('data.txt')
df.head()
```

	Wine	Chips	Bread	Butter	Milk	Apple
0	Wine	Chips	Bread	Butter	Milk	NaN
1	Wine	Chips	Bread	Butter	Milk	NaN
2	Wine	Chips	NaN	NaN	NaN	Apple
3	Wine	Chips	Bread	Butter	Milk	Apple
4	Wine	Chips	NaN	NaN	Milk	NaN

Tạo và đọc file

```
records=[]
for i in range(len(df)):
    records.append([str(df.values[i,j]) for j in range(len(df.columns))])
records
```

Chuyển Dataset về danh sách list

```
def items_count(data, min_support):
    items = {}
    for record in data:
        for item in record:
            if item != 'nan':
                if item not in items:
                    items[item] = 1
                else:
                    items[item] += 1
    frequent_itemset = {}
    for item in items:
        if items[item] >= min_support:
            frequent_itemset[frozenset([item])] = items[item]
    return frequent_itemset
count_items=items_count (records, min_support=0.5)
count_items
```

Đếm số lần lặp lại của từng loại. Tạo items để làm biến đếm. Dùng 2 vòng lặp để đếm từng phần tử. Nếu phần tử được lặp lại thì items +1

```
{frozenset({'Wine'}): 17,
 frozenset({'Chips'}): 15,
 frozenset({'Bread'}): 15,
 frozenset({'Butter'}): 14,
 frozenset({'Milk'}): 16,
 frozenset({'Apple'}): 15}
```

Tính số cặp được lặp lại

```
def pairs_count(dataframe):
# lưu số lượng từng cặp item
    pairs_dict = {}
    for i in range(len(dataframe)):    # Lặp qua các hàng trong dataframe
        for j in range(len(dataframe.columns)):# Lặp qua các cột trong hàng hiện tại
            if isinstance(dataframe.iloc[i, j], str):
                for k in range(j+1, len(dataframe.columns)):
                    if isinstance(dataframe.iloc[i, k], str): # Nếu là item, tạo cặp ite
                        pair = frozenset([dataframe.iloc[i, j], dataframe.iloc[i, k]])
                        pairs_dict[pair] = pairs_dict.get(pair, 0) + 1

    return pairs_dict
```

Lặp từng hàng với cột. Nếu giá trị là item sẽ qua cột khác thực thi để tìm item khác và tăng số cặp đó lên 1

```
itemset_counts =pairs_count(df)
itemset_counts
```

```
{frozenset({'Chips', 'Wine'}): 11,
 frozenset({'Bread', 'Wine'}): 13,
 frozenset({'Butter', 'Wine'}): 11,
 frozenset({'Milk', 'Wine'}): 14,
 frozenset({'Bread', 'Chips'}): 10,
 frozenset({'Butter', 'Chips'}): 10,
 frozenset({'Chips', 'Milk'}): 11,
 frozenset({'Bread', 'Butter'}): 12,
 frozenset({'Bread', 'Milk'}): 12,
 frozenset({'Butter', 'Milk'}): 12,
 frozenset({'Apple', 'Wine'}): 12,
 frozenset({'Apple', 'Chips'}): 10,
 frozenset({'Apple', 'Bread'}): 11,
 frozenset({'Apple', 'Butter'}): 10,
 frozenset({'Apple', 'Milk'}): 11}
```

Tìm các cặp thường gặp minsup = 0.5

```
def frequent_itemsets(itemset_counts, minsup):
    max_count = max(itemset_counts.values())
    minsup_count = int(max_count * minsup)
    #các itemset không phổ biến
    frequent_itemsets = [itemset for itemset, count in itemset_counts.items()
                         if count >= minsup_count]
    return frequent_itemsets
```

```
[frozenset({'Chips', 'Wine'}),
 frozenset({'Bread', 'Wine'}),
 frozenset({'Butter', 'Wine'}),
 frozenset({'Milk', 'Wine'}),
 frozenset({'Bread', 'Chips'}),
 frozenset({'Butter', 'Chips'}),
 frozenset({'Chips', 'Milk'}),
 frozenset({'Bread', 'Butter'}),
 frozenset({'Bread', 'Milk'}),
 frozenset({'Butter', 'Milk'}),
 frozenset({'Apple', 'Wine'}),
 frozenset({'Apple', 'Chips'}),
 frozenset({'Apple', 'Bread'}),
 frozenset({'Apple', 'Butter'}),
 frozenset({'Apple', 'Milk'})]
```

```python
def calculate_confidence(frequent_itemsets, itemset_counts, count_items):
    for itemset in frequent_itemsets:
        items = itemset
        for item in items:
            antecedent = frozenset([item])
            consequent = items - antecedent
            confidence = itemset_counts[itemset] / count_items[antecedent]
            print(f"conf({set(antecedent)},{set(consequent)}) = {confidence:.3f}")
```

Tính độ tin cậy confidence với công thức đã cho trong hướng dẫn

```
conf({'Wine'},{'Chips'}) = 0.647
conf({'Chips'},{'Wine'}) = 0.733
conf({'Wine'},{'Bread'}) = 0.765
conf({'Bread'},{'Wine'}) = 0.867
conf({'Wine'},{'Butter'}) = 0.647
conf({'Butter'},{'Wine'}) = 0.786
conf({'Wine'},{'Milk'}) = 0.824
conf({'Milk'},{'Wine'}) = 0.875
conf({'Bread'},{'Chips'}) = 0.667
conf({'Chips'},{'Bread'}) = 0.667
conf({'Butter'},{'Chips'}) = 0.714
conf({'Chips'},{'Butter'}) = 0.667
conf({'Chips'},{'Milk'}) = 0.733
conf({'Milk'},{'Chips'}) = 0.688
conf({'Butter'},{'Bread'}) = 0.857
conf({'Bread'},{'Butter'}) = 0.800
conf({'Bread'},{'Milk'}) = 0.800
conf({'Milk'},{'Bread'}) = 0.750
conf({'Butter'},{'Milk'}) = 0.857
```

```
association_rules = apriori ( records , min_support = 0.50 , min_confidence = 0.7 , min_lift = 1.2 )
association_results = list ( association_rules )
```

```
def inspect ( output ) :
    lhs        =  [ tuple ( result [ 2 ] [ 0 ] [ 0 ] ) [ 0 ] for result in output ]
    rhs        =  [ tuple ( result [ 2 ] [ 0 ] [ 1 ] ) [ 0 ] for result in output ]
    support    = [ result [ 1 ] for result in output ]
    confidence = [ result [ 2 ] [ 0 ] [ 2 ] for result in output ]
    lift       = [ result [ 2 ] [ 0 ] [ 3 ] for result in output ]
    return list ( zip ( lhs , rhs , support, confidence , lift ) )
```

```
output_DataFrame = pd.DataFrame(inspect(association_results),
                   columns = ['Left_Hand_Side','Right_Hand_Side','Support','Confidence','Lift'])
```

```
output_DataFrame
```

Chạy đoạn code với ariori như hướng dẫn

	Left_Hand_Side	Right_Hand_Side	Support	Confidence	Lift
0	Bread	Butter	0.571429	0.8	1.2

Nhận thấy confidence giữa 2 phương pháp đều là 0.8